Towards Dynamic Context Discovery and Composition

Graham Thomson, Sotirios Terzis, and Paddy Nixon
The Global and Pervasive Computing Group
Department of Computer and Information Sciences
The University of Strathclyde
Glasgow, Scotland.
e-mail: {firstname.lastname}@cis.strath.ac.uk

***Abstract*— Context-awareness has been identified as a key characteristic for pervasive computing systems. As a variety of context-aware environments begin to flourish, pervasive applications shall have to interact different environments well. In this paper we propose extensions to the Strathclyde Context Infrastructure that gives context-aware applications the potential to adapt to unfamiliar environments transparently. We present a vision of a context discovery technique based on automated semantic reasoning about context information and services. The technique will offer higher levels of scalability and of interoperability with new context environments that cannot be achieved with current methods.**

I. Introduction

In [1], the ability for applications to be context-aware is identified as a key characteristic for pervasive computing. However, context-aware computing systems are still at an early stage of research. They are small systems - they perhaps cover a few rooms at most. They are likely to be developed by a single research team who work in close collaboration. The number and variety of context entities may be few. Future context-aware computing systems can assume none of these qualities.

As a variety of context-aware environments begin to flourish, pervasive applications shall have to interact with environments they have never encountered before that although may be unfamiliar in appearance, are semantically similar to environments they have encountered before. In this paper we propose extensions to the Strathclyde Context Infrastructure (SCI) [2] that gives context-aware applications the potential to adapt to these new environments transparently.

In the next section, we go on to look briefly at context and how it is represented in the SCI. Following this, related work in open distributed systems is described, before going on to present our extension to this work to achieve semantic context trading.

II. Context, Discovery and the Strathclyde Context Infrastructure

A. Dey describes context-aware computing systems in the following manner:

> "a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task"[3]

Furthermore, he and others [4][5][6][7] identify context discovery as a key element of the operation of context-aware applications.

In this paper we present a vision of a context discovery technique based on automated semantic reasoning about context information and services. The technique will offer higher levels of scalability and of interoperability with new context environments that cannot be achieved with current methods. We also describe how this new approach could be integrated into the Strathclyde Context Infrastructure (SCI), which is described next.

For a full description of the SCI, please refer to [2]. Here, only a brief description is given that is sufficient for understanding the concepts discussed in this paper.

The main element of the SCI is the Range. A Range is a bounded physical or logical area. Each Range contains a single Context Server, which manages three types of components; Context Entities, Context Utilities and Context Aware Applications.

The Context Server (CS) is the most important component of a Range. It manages the other components and provides the means of communicating with other Ranges in a network.

A Context Entity (CE) is a lightweight software component for representing an entity within the infrastructure. A CE maintains a Profile for its entity that contains meta-data describing the entity. For entities that provide a service, the CE may also maintain an Advertisement describing the services that this entity can provide to other entities.

A Context Aware Application (CAA) is an application that has the ability to pull or be pushed contextual information to or from the infrastructure. A CAA communicates with the CS by way of a Query, which is represented as a short XML document.

The Context Utilities (CU) are a set of specialist services that help the CS in the management of a Range. These include a Range Service which is responsible for detecting the arrival into and departure of entities from a Range, and a Query Resolver which provides the means to take a high level query and decompose it into a useful configuration of Context Entities.

III. Semantically Enhanced Component Trading

The ideas for the extensions to the SCI are inspired by previous work on semantic component trading in open distributed systems [8].

When using a trader service, such as the CORBA Trader Service [9], a client requests a service type - a description of an interface and a set of desired properties - and the trader returns a set of references to object implementations that match the interface type and satisfy the set of properties. These object implementations are referred to as service *offers*.

Two aspects of this current approach are too limited for component trading [8]: the interface type definition describes only the syntactic characteristics of services, and matching of these types is based solely upon a syntactically defined notion of type conformance.

Semantically enhanced component trading [8] overcomes these limitations by augmenting component type descriptions with specifications of required interfaces and a behavioural specification of the component itself. Using this additional information, the resulting set of component offers is improved in the following two ways. First, the precision of the result is increased as we have stronger assurances that what we receive is what we asked for. Secondly, a rich set of semantic relationships is constructed that is exploited by the trader by substituting compatible types, which may be obtained through component composition. This greatly increases the level of recall in the result.

IV. Semantically Enhanced Context Trading

For the SCI, we wish to have models for context that abstract away from the details of individual sensors and sensor types. This would allow contextualised services to utilise context information from a large variety of sensors of differing types [6]. To achieve this however, we must address the issues of how an application will locate the appropriate set of sensors in its operating environment, and how the application can compose different sensors to produce the required context information.

Current research into these issues is inspired by ideas from distributed systems research (see [3] for example) and focus on common basic mechanisms like naming, directory services or simple trading services. We wish to harness the power of semantic component trading in tackling these issues in context-aware computing systems.

It is easy to imagine a context entity as a software component. In fact, we could consider the problem of discovering and composing the appropriate context entities to be a special case of the more general problem of discovering and composing components in software engineering [10]. This allows us to apply the semantic component trader concepts directly.

Currently in the SCI, context information is represented as a configuration of context entities. What we hope to achieve is to be able to express requests for context information at an abstract level, that is, in terms of basic context elements and context operators instead of concrete context entities and configurations. Basic context elements would be an abstraction of context entities, and context operators structures for composition of basic context elements to build higher-level contexts.

With requests formed in this way, the semantic context trader can then search for compatible basic elements and operators and form all equivalent context expressions. Then after examining the set of available context entity offers, determine which equivalent context expressions can be instantiated. These instantiations would then be passed to a context composition mechanism to construct the resulting set of configurations.

In this way, a context-aware application can move from one range to another and have the query resolver deliver configurations formed from context entities from the new range dynamically and at the greater levels of precision and recall the semantic trader offers.

After experimenting with this modified architecture we hope to be able to identify an appropriate set of basic context elements that deliver the required context information, and a satisfactory set of context operators.

V. Summary

We have presented our ideas for a technique for trading in context-aware computing systems. We described an approach to combat the problems of scalability and interoperability that open distributed systems suffer from. Fortunately, we are in a position that we do not need to wait for context-aware computing to reach the necessary level of maturity to hit this ‘semantic barrier’; here, we can tackle this problem at its inception.

References

[1] T. Kindberg and A. Fox, “System software for ubiquitous computing,” *IEEE Pervasive Computing*, vol. 1, no. 1, 2002.

[2] R. Glassey, G. Stevenson, M. Richmond, F. Wang, P. Nixon, S. Terzis, and I. Ferguson, “Towards a middleware for generalised context management.” in *1st International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC03)*, June 2003.

[3] A. Dey, “Providing architectural support for building context-aware application,” Ph.D. dissertation, Georgia Institute of Technology, Atlanta, 2000.

[4] K. C. N. Davies, K. Mitchell and G. Blair, “Developing a context-sensitive tour guide,” in *1st Workshop on Human Computer Interaction for Mobile Devices*, 1998.

[5] J. Coutaz and G. Rey, “Foundations for a theory of contextors,” 2002.

[6] A. S. H.W. Gellersen and M. Beigl, “Multi-sensor context-awareness in mobile devices and smart artefacts.”

[7] P. Gray and M. Sage, “Dynamic links for mobile connected context-aware systems,” 2001.

[8] S. Terzis and P. Nixon, “Component location and the role of trading in large scale distributed systems,” 2000.

[9] O. M. Group, “Corba trading object service specification,” 2000.

[10] F. M. H. Mili and A. Mili, “Reusing software: Issues and research directions,” *IEEE Transactions On Software Engineering*, vol. 21, no. 6, 1995.